club commodore

Boletín informativo para los usuarios de microordenadores VIC y CBM





enero 1983

CLUB COMMODORE 1982

Para celebrar el cambio de año publicamos la portada de los números que han salido hasta ahora — cosa que suelen hacer las revistas con solera, como la nuestra —, y para seguir la tradición debemos anunciar (no sabemos aún si con satisfacción o tristeza) que hemos agotado los ejemplares de los números 0 y 1 y que del 2 quedan pocos. Como en nuestra Revista no queremos que se desperdicie nada, pretendemos que esto sea un índice informal del contenido de CLUB COMMODORE. ¡Ahí va!

club commodore
Boletín informativo para los usuarios de microordenadores
VIC y CBM
una aportación al fomento de la microinformática (pág. 1)
ventana CBM: el acceso a lo «mejor» de los Commodore (pág. 2)
conversión sencilla de decimal a «hex» y al revés (pág. 3)
mapa de memoria del VIC-20 (pág. 4)
programa para el cálculo de atenuadores en Pi o en T (pág. 6)
correo abierto (pág. 8)
Nº/0 septiembre 1982

club commodore
Boletín informativo para los usuarios de microordenadores
VIC y CBM
VIC-20: algunos "pokes" interesantes (pág. 1)
un calendario perpetuo (pág. 2)
ventana CBM: protección de programas inhibición de la tecla "stop" (pág. 3)
un programa de utilidad (pág. 3)
mapa de memoria del VIC-20: una ojeada al Video Interface Chip (pág. 4)
marketclub (pág. 7)
Nº/1 octubre 1982



NÚMERO 0

NÚMERO 1

NÚMERO 2

NÚMERO 3



Depósito Legal Sep. B. 2133-1958 - Altés, s.l., Caballero 87, Barcelona

EDITORIAL

a nuevos tiempos, nuevos...

...Precios. Nuestra Revista del CLUB COMMODORE ha crecido sustancialmente a partir del número de diciembre. Ha doblado su contenido pasando, para los suscriptores, de tener ocho páginas *exactamente a tener* dieciséis, *con lo que los costes de confección se han disparado y por ello hemos tenido que tomar una decisión harto dolorosa: subir el precio de suscripción que pasará a ser ahora de 1.980 ptas. los once números anuales, con lo que el precio del ejemplar queda en 180 ptas.*

Imaginamos perfectamente la opinión del lector sobre esta tema. Solamente queremos hacer constar que, como usuarios que somos, los que componemos la Redacción de CLUB COMMODORE comprendemos perfectamente sus quejas. Quede para consuelo de los más afortunados el hecho de que para los suscriptores actuales esta subida no tendrá efecto hasta que se cumpla el aniversario de su suscripción.

Además, nos comprometemos a no dormirnos y a trabajar intensamente en la mejora de nuestra Revista y los servicios a ella asociados. En este sentido nuestros suscriptores pertenecientes al CLUB COMMODORE están recibiendo ya, durante tres meses y gratuitamente, el semanario "COMPUTERWORLD".

¡Hasta el próximo número! ■

VENTANA CBM

la zona de «strings» (2.ª parte) y el «garbage collection»

por **J. C. SAMARANCH**

El mes pasado empezamos a comentar el comportamiento de la zona de cadenas de caracteres (strings), pero existen varias **curiosidades** de las cuales, seguramente, no nos hemos percatado en el uso cotidiano del sistema o, en otros casos, hemos dado la culpa de anomalías acaecidas, sin sentido aparente, al duende de la informática (BUG en nuestro caso).

Como es habitual, ilustraremos la explicación con un programa sencillo:

```
20 C$ = «COMMODORE»
50 A = PEEK (48) + PEEK (49) * 256
60 B = PEEK (52) + PEEK (53) * 256
70 PRINT B-A
```

Al efectuar el RUN nos imprime un CERO correspondiente a la longitud de la zona de «strings» (B — A = principio — final). Parece extraño ¿no? En la edición anterior creamos la misma variable y esta zona ocupaba 11 bytes! (Ver CLUB COMMODORE núm. 3.)

Si sustituimos la línea 20 por: 20 C$ = «COMMODORE "+"», solamente hemos añadido NADA, que no debería tener importancia, y ejecutamos el RUN, el resultado en este caso es: 11 bytes, tal y como comentábamos.

¿Qué ha sucedido? En ambos casos, se ha creado una cabecera para la variable C$ en la zona de variables pero dicha cabecera «apuntaba» (indicaba dónde se hallaba almacenada la cadena correspondiente a C$) a distintos sitios.

En el primer caso, apuntaba hacia la zona de programas, exactamente al punto donde se encontraba el «string» (en la línea 20 después de C$ = »). Esto se hace para descargar de cadenas la zona de «strings» que es donde se sitúa en el segundo caso por interpretar que se trata de una composición de cadenas (signo +).

Si añadimos al programa anterior las siguientes líneas:

```
10 FORI = 1T0100
30 NEXT
```

En el primer caso, nos continúa dando longitud CERO porque siempre definimos la misma variable que se encuentra dentro del programa y que no se trata de una fórmula. En cambio, en el segundo caso la longitud es 1100, como si hubiéramos creado 100 cadenas distintas de 11 bytes cada una.

Efectivamente cuando se redefine un «string» situado en la zona de «strings» no sustituye al anterior sino que se crea uno nuevo y se destruyen los links (punteros) del anterior a la variable, situada en la zona de variables.

(pasa a la pág. siguiente)

AVISO PARA NAVEGANTES

● Aunque a primera vista esta sección está dedicada de manera específica a los ordenadores CBM (actualmente se comercializan las series 4000 y 8000), el contenido de los artículos es completamente aplicable — con ligeras modificaciones que se hacen constar en los propios artículos — a equipos como el 2001, el 3000 e incluso el VIC-20, dado que el funcionamiento de los diferentes intérpretes de BASIC es sustancialmente el mismo. Cambiando, como máximo, el valor de los POKEs y PEEKs, se pueden conseguir resultados. En todo caso, los suscriptores de CLUB COMMODORE disponen de los correspondientes mapas de memoria que se están publicando en forma de fichas en la contraportada interior de la Revista a partir del número 1.

COLABORACIONES

un juego de inteligencia

por ERNESTO MARTÍNEZ DE CARVAJAL HEDRICH

VENTANA CBM

(viene de la pág. anterior)

EL «GARBAGE COLLECTION»

Como hemos visto, tenemos 100 cadenas en la zona de «strings» y una sola variable válida. Por lo tanto, las 99 cadenas «malas» es lo que se denomina **garbage** (basura).

Si activamos el proceso de «garbage collection» (literalmente recogida de basura) se limpia la zona de «strings» y se recupera memoria libre.

Si aplicamos la siguiente fórmula:

PRINT PEEK (48) + PEEK (49) * 256 — PEEK (46) — PEEK (47) * 256

obtenemos el número de bytes libres en este momento. (Ver mapa en la edición anterior.) Si escribimos:

PRINT FRE (0)

obtenemos el número de bytes libres después de ejecutarse el «garbage collection» (provocado por la instrucción FRE (0)), siendo, por tanto, mayor (o igual, en el peor de los casos) que el valor obtenido anteriormente.

Si repetimos ahora la primera fórmula observaremos que la memoria libre se ha incrementado y es igual al valor del FRE (0).

En nuestro caso concreto estos valores son: 30519 y 31608, respectivamente. La diferencia es, precisamente, 99 variables «malas» × 11 bytes cada una = 1089. ■

A LOS USUARIOS DE "VIC" DE BARCELONA

● Se necesitan colaboradores para redacción y traducción de programas para el «VIC». Es imprescindible ser usuario y residir en Barcelona. Los interesados pueden llamar a: P. Masats, «Microelectrónica y Control, S. A.» - Taquígrafo Serra n.° 7 - 5.ª planta - Teléfono 250 51 03, en horas de oficina. - Barcelona.

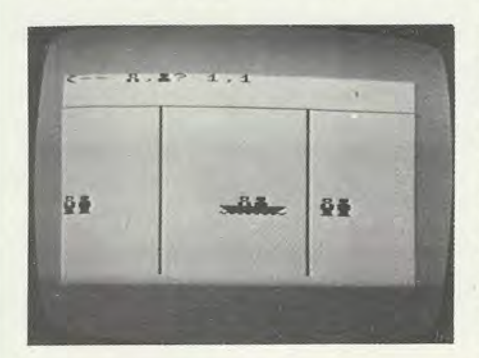
Conocimos a Ernesto en medio de los avatares de una feria (SONIMAG) y en medio de todo el barullo nos enseñó algunos de sus programas (que también vende) para el VIC. Nos gustaron y ahora publicamos una colaboración suya que esperamos sea la primera de una larga serie... Como ya se presenta él solo no vamos a dar más la lata.

Apreciados convicciosos:

Ante todo, felicidades a los culpables de nuestra Revista que, sin ánimos de grandeza, nos informa, nos ayuda, y, sobre todo, nos une.

En esta ocasión quisiera aportar mi grano de arena con dos programas, un juego y una rutina que espero os divierta y sirva de ayuda.

Antes de proceder a explicaros los programas voy a presentaros a mi sistema microinformático que, como podéis ver en la foto, va «sobre ruedas».





Para evitar el enredo de cables y el engorro que supone tener todos los componentes desperdigados sobre una mesa, se me ocurrió esta sencilla y práctica solución. Básicamente se trata de una consola de madera (aún no existía la que en la actualidad comercializa Commodore) y una mesa de máquina de escribir. Por unas nueve mil pesetas adquirí todo el material necesario, la mesa, el aglomerado de madera laminado en «formica» blanca, un pequeño ventilador, cable eléctrico, clavos, etc... Y en una lluviosa tarde de domingo dejé lista la obra.

La consola está formada por una base sobre la que se apoya el VIC y un receptáculo en el cual encaja la parte posterior de aquél, de manera que los cables quedan ocultos en el interior. La parte superior, sobre la que descansa el monitor de TV y el cassette, se puede levantar para acceder a su interior, en el cual se encuentra, además de la maraña de cables, el transformador, el modulador y el ventilador que refrigera todo el conjunto, aspirando aire a través de unos orificios practicados en la base. De la parte posterior sale un ¡único! y largo cable para la toma de corriente y un enchufe con tres salidas para conectar los aparatos que no se alimentan directamente del VIC. Un pequeño interruptor situado en la parte derecha, conecta todos los elementos. La consola no está fijada a la mesa para permitir su traslado independientemente de ésta. Cuatro tacos de goma, fijados a la base, impiden que pueda deslizarse, a la vez que evita rayar la superficie sobre la que pongamos la consola. En la mesa, tres tapas laterales y otros tantos estantes regulables completan el conjunto. Práctico, ¿no?

Y después de los trabajos manuales, pasemos a la informática que es lo nuestro. El primer programa es un popular juego conocido con el nombre de «Misioneros y Caníbales» aunque haya otras versiones similares con diferentes nombres. El objetivo del juego es trasladar un grupo de tres misioneros y otros tantos caníbales que se encuentran en la orilla de un río, a la opuesta, utilizando para ello una barca en la que sólo caben dos personas. Sólo una regla: si en algún momento hay inferioridad de misioneros en alguna

PROGRAMA NÚM. 1

```
20 G1=32768
30 G2=7168
40 IP=7680
50 PRINT "♥":GOSUB2000
60 FOR I=0 TO 512
70 K=PEEK(G1+I)
80 POKE (G2+I),K
90 NEXT I
1000 DATA 0,60,36,36,24,36,102,102,102
1010 DATA 1,60,60,60,24,60,126,126,126
1020 DATA 2,126,126,60,60,60,60,126,0
1030 DATA 3,255,127,63,31,15,159,102,0
1040 DATA 4,255,255,255,255,255,255,102,
0
1050 DATA 5,255,254,252,248,240,249,102,
0
1060 DATA 6,189,165,165,153,231,36,36,36
1070 DATA 7,255,126,60,126,255,255,126,6
0
1080 DATA 8,0,0,0,0,0,0,0,255
1090 DATA 9,32,32,32,32,32,32,32,32
1510 FOR I=1 TO 10
1515 READ C
1520 FOR J=0 TO 7
1530 READ K
1540 POKE G2+(C*8)+J,K
1550 NEXT J
1560 NEXT I
1600 POKE 52,28:POKE 56,28
1610 CLR
1620 PRINT"PULSE ■SHIFT■&■RUN/STOP■"
1630 STOP
2000 POKE 36878,15
2005 PRINT "■  ERNESTO MARTINEZ DE"
2010 PRINT"■   CARVAJAL HEDRICH"
2015 POKE 36874,210
2016 FORI=1TO1000:NEXT
2020 PRINT"■■■          PRESENTA"
2022 POKE 36875,180
2023 FORI=1TO1000:NEXT
2025 PRINT"■■■    ****************"
2027 PRINT"    *               *"
2130 PRINT "■■■■■ MISIONEROS *"
2140 PRINT"    *       Y       *"
2150 PRINT"■■■■■ ■CANIBALES■  *"
2155 PRINT"    *               *"
2160 PRINT"    *****************"
2170 POKE 36876,128
2180 FORI=1TO2000:NEXT
2190 FOR I=10TO0STEP-1:POKE36878,I:FORJ=
1TO500:NEXT:NEXT
2200 POKE 36874,0:POKE36875,0:POKE36876,
0
2300 RETURN
4000 FOR I=0TO64
4010 PRINTI;"=";CHR$(I+64)
4020 NEXTI

READY.
```

PROGRAMA NÚM. 2

```
110 M(1)=3:M(2)=0:M(3)=0
120 N(1)=3:N(2)=0:N(3)=0
130 I$(5)="CDDE":B=8
140 B9$="      "
150 M$="@@@@@@"
160 N$="AAAAAA"
170 B$="BBBBBB"
200 POKE 36869,255
210 PRINT"♥"
220 GOSUB1600:GOTO3010
1600 I1$=B9$+"I"+B9$+"   "+"I"+B9$
1610 I2$=B9$+"I"+B9$+"   "+"I"+B9$
1620 I1$=LEFT$(M$,M(1))+LEFT$(N$,N(1))+R
IGHT$(I1$,22-(M(1)+N(1)))
1630 I1$=LEFT$(I1$,B)+LEFT$(M$,M(2))+LEF
T$(N$,N(2))+RIGHT$(I1$,22-(B+M(2)+N(2)))
1640 I2$=LEFT$(I2$,B-1)+"CDDE"+RIGHT$(I2
$,22-(B+3))
1650 I1$=LEFT$(I1$,16)+LEFT$(M$,M(3))+LE
FT$(N$,N(3))+RIGHT$(I1$,22-(16+M(3)+N(3)
))
1660 I2$=LEFT$(B$,M(1)+N(1))+RIGHT$(I2$,
22-(M(1)+N(1)))
1670 I2$=LEFT$(I2$,16)+LEFT$(B$,M(3)+N(3
))+RIGHT$(I2$,22-(16+M(3)+N(3)))
2005 PRINT"■"
2010 PRINT"                      "
2015 PRINT"                      "
2020 PRINT"HHHHHHHHHHHHHHHHHHHHHH";
2030 FORI=1TO10
2040 PRINTB9$;"I          I";B9$;:NEXTI
2060 PRINTI1$;
2070 PRINTI2$;
2080 FORI=1TO6:PRINTB9$;"I          I";B9$
;:NEXTI
2200 RETURN
3010 SW=0
3020 GOSUB6000:GOSUB1600
3040 FORB=8TO12:GOSUB1600:NEXTB:B=12
3045 M(3)=M(3)+M(2):M(2)=0:N(3)=N(3)+N(2
):N(2)=0:GOSUB1600:GOSUB3510
3060 SW=1:GOSUB6000:GOSUB1600
3070 FORB=12TO8STEP-1:GOSUB1600:NEXTB:B=
8
3075 M(1)=M(1)+M(2):M(2)=0:N(1)=N(1)+N(2
):N(2)=0:GOSUB1600:GOSUB3510
3080 GOSUB 3510
3090 GOTO3010
3510 IFM(1)<N(1)ANDM(1)>0THENSW=0:GOTO50
10
3520 IFM(3)<N(3)ANDM(3)>0THENSW=1:GOTO50
10
3530 IF M(1)=0ANDN(1)=0THEN4010
3540 RETURN
4010 Z$="BRAVO, LO CONSEGUISTE":GOTO5090
5010 K=1:J=6:IFSW=1THENK=16:J=22
5020 FORI=KTOJ
5030 IFMID$(I1$,I,1)<>"@"THEN5060
5040 I1$=LEFT$(I1$,I-1)+"F"+RIGHT$(I1$,2
2-(I))
5050 I2$=LEFT$(I2$,I-1)+"G"+RIGHT$(I2$,2
2-(I))
5060 NEXTI:GOSUB2005
5080 Z$="   JA, JA PERDISTE"
5090 FORI=1TO4000:NEXTI
5100 POKE 36869,240
5110 PRINT"■■■■■■■■■■■";Z$
5500 GETA$:IFA$=""THEN5500
5510 RUN
6000 PRINT"■                    ■";
6020 M=0:C=0
6030 ON SW+1 GOTO6040,6050
6040 INPUT "--> @,A";M,C:GOTO 6100

6050 INPUT "<-- @,A";M,C
6100 IFSW=1THENIFM>M(3)ORC>N(3)THEN6200
6110 IF M<0ORM>2ORC<0ORC>2THEN6200
6140 IF M+C>2ORM+C=0THEN6200
6150 IFSW=0THENM(1)=M(1)-M:N(1)=N(1)-C
6160 IFSW=1THENM(3)=M(3)-M:N(3)=N(3)-C
6170 N(2)=N(2)+C
6180 M(2)=M(2)+M
6190 RETURN
6200 POKE 36878,15:POKE 36875,200
6210 FORI=1TO200:NEXTI
6220 POKE 36878,0:POKE 36875,0:GOTO6000

READY.
```

PROGRAMA NÚM. 3

```
1 REM DE LA LINEA
2 REM 1 A LA 9999
3 REM ES EL PROGRAMA
4 REM A RENUMERAR
5 REM *****************
6 REM ATENCION!
7 REM SI SE TRABAJA CON
8 REM EL CARTUCHO DE 3K
9 REM EL VALOR DE R6 EN
10 REM LA LINEA 10010
11 REM DEBE SER 1024;
12 REM SI SE TRABAJA
13 REM CON EL DE 8 O 16K
14 REM R6 DEBE SER IGUAL
15 REM A 4608
9999 END
10000 REM RENUMBER
10010 R6=4096:R7=10
10020 IFPEEK(R6+3)=6ANDPEEK(R6+4)=39THEN
END
10030 R8=INT(R7/256):R9=R7-256*R8:POKER6
+3,R9:POKER6+4,R8
10040 IFPEEK(R6+5)<>0THENR6=R6+1:GOTO100
40
10050 R7=R7+10:R6=R6+5:GOTO10020

READY.
```

de las dos orillas, los caníbales los meten en la olla y... se los comen.

El programa está partido en dos. El primero traslada la tabla de generación de caracteres, creando aquellos especiales que necesita el segundo. No detallo más este capítulo ya que no es evidente y escapa a las pretensiones de este artículo. El segundo programa es el que realiza el juego propiamente dicho. En la pantalla aparecen dos líneas verticales que representan los márgenes de río, el grupo de misioneros y caníbales que, en un principio estaban a la izquierda, así como la barca. En la parte superior aparece una representación formada por una flecha que indica el sentido en

(pasa a la pág. siguiente)

un juego de inteligencia

(viene de la pág. anterior)

que se va a realizar el siguiente traslado. El número de personajes a trasladar de cada tipo se ha de dar por dos números separados por una coma. En caso de que no se pueda trasladar algún tipo, se ha de poner un cero.

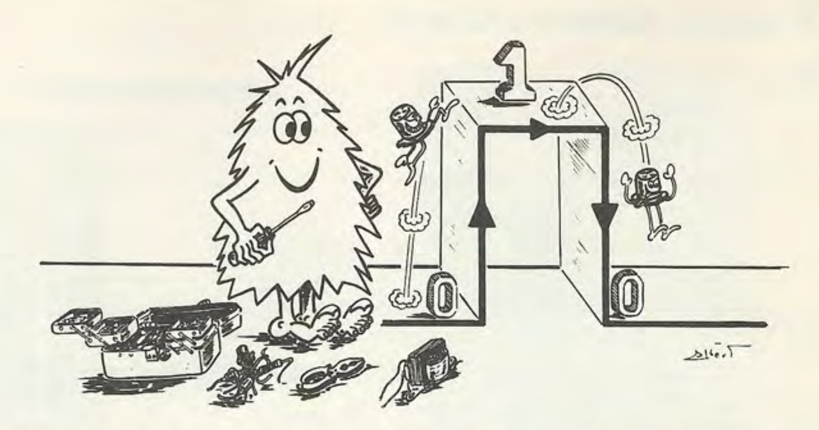
El programa puede ser mejorado en los siguientes puntos:

— Forma de pedir los personajes a trasladar.
— Colores.
— Número de personajes variable (aunque siempre ha de ser igual el número de misioneros al de caníbales).

Si sólo se dispone de tres K, no se puede añadir prácticamente ni un byte más por lo que habrá que tener cuidado al realizar la transcripción, y, si queréis mejorar el programa, tendréis que ampliar la memoria teniendo en cuenta que habrá que modificar los peek y pokes que gestionan la nueva tabla de caracteres.

Y, por último, la rutina de utilidad que, con seguridad, os ahorrará mucho tiempo en el desarrollo de programas. Se trata de un renumerador de líneas o resecuenciador. Como podéis ver se trata de un programa muy pequeño, que ocupa sólo 198 bytes. Para utilizarlo basta entrarlo a partir de la línea 10000, teniendo en memoria el programa a renumerar (evidentemente, éste no puede tener ninguna instrucción más allá de esa línea), y hacer «run 10000». Empezando en la instrucción 10, se va incrementando las siguientes en diez unidades, hasta encontrar el «end» del programa. Los números de sentencia que aparecen en las instrucciones «GOTO» y «GOSUB», no se modifican, debiéndose hacer después manualmente. Para ello, antes de realizar la renumeración, añadir al final de dichas instrucciones y de las que a éstas hacen referencia, un REM que las relacione. Tras la renumeración, se modificarán dichas instrucciones manualmente, eliminando los REMs.

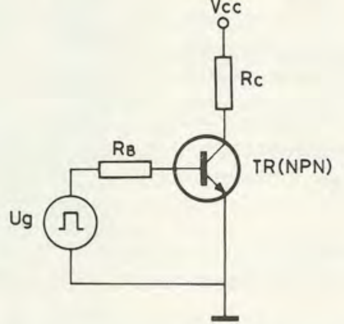
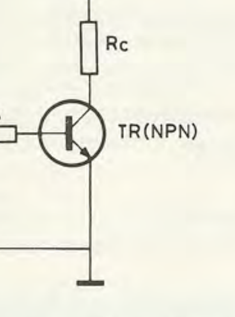
La base teórica de esta rutina está en la forma en que son almacenadas en RAM las instrucciones BASIC, que es la siguiente: los dos primeros bytes encadenan cada línea con su siguiente, los dos siguientes son el número de sentencia, a continuación está la instrucción BASIC y, por último, un NULL. En la última línea hay dos NULLs más que indican el final del programa. ■



Este programa nos ayuda a diseñar circuitos de conmutación con transistores NPN de silicio.

Los datos que nos pide el programa son:

— HFE MIN: valor mínimo de hFE del transistor.
— ICBO (mA): intensidad de corte colector-base.
— UG + (V): valor máximo de tensión positiva de entrada.

— UG — (V): valor máximo de tensión negativa de entrada.
— Ucc (V): tensión de alimentación del circuito.

Todos los resultados vienen dados en:

— miliamperios si se trata de intensidades.
— en kilo ohmios si se trata de resistencias.

FUNCIONAMIENTO

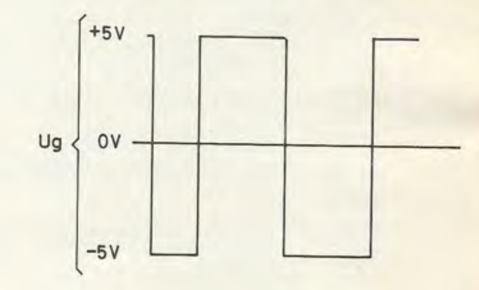
Las líneas 10-20 hacen la presentación del programa. En las líneas 30-70 se efectúa la entrada de datos de las variables del programa con sus correspondientes unidades. Las líneas 80-120 son para el cálculo del estado de saturación y la representación en la pantalla de estos datos.

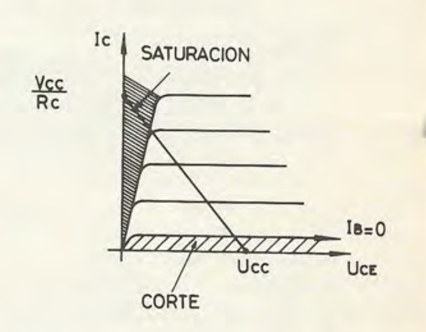
Las líneas 210-240 realizan la normalización de las resistencias dadas por el cálculo.

En las líneas 250-320 se realiza el proceso inverso del proyecto para diferenciar el resultado teórico del normalizado. En las líneas 330-370 se presentan los resultados.

De 380 a 430 se encuentra una rutina para seguir el proyecto o corregir datos en el caso de que éstos no se aproximen a lo calculado en el proceso teórico.

De 440 a 470 se efectúa la presentación en pantalla de los resultados finales del proyecto.





En 480-540 se encuentra una rutina de fin de programa y entre 550 y 570 se realiza el paso de una pantalla a otra.

Las fórmulas utilizadas en el cálculo son:

$IB = Icbo$
$RB = -UG/IB$
$UCE = .1$
$IB = (UG-.7)/RB$
$IB > IC/hfemin$
$IC = (UCC-UCE)/RC$

diseño de circuitos con transistores en régimen de conmutación

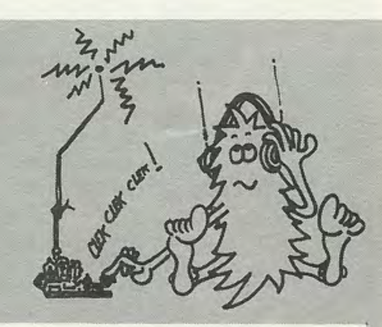
por **J. JOSÉ AGUILERA** y **R. PARDO**

PROGRAMA

```
0 rem conmutacion
10 rem r.Pardo j.j. aguilera
20 Print"■"tab(3)"■*Conmutacion*
30 Print:inPut"hFEmin";hm
40 Print:inPut"Icbo(mA)";i1
50 Print:inPut"Ug+(V)";u1
60 Print:inPut"Ug-(V)";u2
70 Print:inPut"Ucc(V)";u3
80 Print"■Estado de corte"
90 rb=u2/i1
100 Print:Print"Rb=";rb;"Kohms"
110 gosub 550
120 Print"■"
130 Print"Estado de saturacion"
140 ib=(u1-.7)/rb
150 ic=(hm*ib)/1.5
160 rc=(u3-.1)/ic
170 Print:Print"Ib=";ib;"mA"
180 Print:Print"Ic=";ic;"mA"
190 Print:Print"Rc=";rc;"Kohms"
200 gosub550
210 Print"■Normalizacion"
220 Print"Rb=";rb;"Kohms":Print"Rc=";rc;
"Kohms"
230 Print:Print"Normalice valores"
240 Print:inPut"Rb";rv:inPut"Rc";rz:Prin
t"■"
250 Print"Variacion estado de corte"
260 in=u2/rv
270 Print"Ib(dada)=";i1;"mA"
280 Print"Ib(calculada)=";in;"mA"
290 Print"Variacion estado saturacion"
300 il=(u1-.7)/rv
310 i3=(hm*in)/1.5
320 i2=(u3-.1)/rz
330 Print"Ib(dada)=";ib;"mA"
340 Print"Ib(calculada)=";il;"mA"
350 Print"Ic(dada)=";ic;"mA"
360 Print"Ic(calculada)=";i2;"mA"
370 gosub550
380 Print"■1-Seguir el Proyecto"
390 Print:Print"2-Corregir datos"
400 Print:Print"Pulse 1 o 2"
410 geta$:ifa$="1"then440
420 ifa$="2"then210
430 goto410
440 Print"■Resultados finales"
450 Print:Print"Rc=";rz;"Kohms"
460 Print:Print"Rb=";rv;"Kohms"
470 gosub550
480 Print"■1-Repetir resultados"
490 Print:Print"2-Nuevo Proyecto"
500 Print:Print"3-Fin de Proyecto"
510 Print:Print"Pulse 1,2 o 3"
520 getb$:ifb$=""then520
530 m=val(b$):onmgoto80,20,540
540 end
550 Print:Print"Pulse una tecla"
560 getc$:ifc$=""then560
570 return

ready.
```

RADIOAFICIÓN



cálculo de distancias entre QTH locator

El programa que presentamos hoy es una adaptación para el VIC-20 de otro que nos facilitó EA3BRA para calcular la distancia en kilómetros entre el QTH Locator del usuario y los de los distintos corresponsales. Ha sido utilizado en concursos de VHF para el cómputo de los «logs», ya que dispone, además, de un totalizador de puntos (1 punto = 1 Km.).

En la variable LS de la línea 60 debe figurar nuestro QTH Locator que se tomará como referencia (p. e.: el BB31J de EA3BRA corresponde a Sabadell). A partir de este momento, nos irá pidiendo los distintos QTH's, calculando y visualizando las distancias y acumulando el total de puntos. Cuando no tengamos más para entrar escribimos «←» (flecha horizontal) y RETURN mostrándose por pantalla el total conseguido.

```
1 REM *** CALCULO DE DISTANCIAS ***
2 REM *** ENTRE DOS QTH LOCATOR ***
3 REM *** POR EA3BRA ***
10 DATA3,1,1,1,1,3,1,5,3,5,5,5,5,3,5,1,3
,1,3,3
20 FORI=0TO9:READH(I),K(I):NEXT:T=0
30 DIMV(25):DATA0,1,2,3,4,5,6,7,8,9,10,1
1,12
40 DATA-13,-12,-11,-10,-9,-8,-7,-6,-5,-4
,-3,-2,-1
50 FORI=0TO25:READV(I):NEXT:R=π/180
60 L$="BB31J":REM *** QTH LOCATOR DE ORI
GEN ***
65 GOSUB100:GOSUB160:X0=X*R:Y0=Y*R
70 INPUTL$:IFL$="←"THEN230
75 GOSUB100:GOSUB160:X=X*R:Y=Y*R
80 L=SIN(Y)*SIN(Y0)+COS(Y)*COS(Y0)*COS(X
-X0)
90 L=ATN(SQR(1-L*L)/L)/R*111.323:T=T+L:P
RINTL"KMS.":GOTO70
100 REM *** CALCULO DE LAS LONGITUDES **
*
110 A=V(ASC(MID$(L$,1,1))-65)
120 D=VAL(MID$(L$,4,1))
130 IFD=0THEND=10
140 H=H(ASC(MID$(L$,5,1))-65)
150 X=2*A+D/5-H/30:RETURN
160 REM *** CALCULO DE LAS LATITUDES ***
170 B=V(ASC(MID$(L$,2,1))-65)
180 C=VAL(MID$(L$,3,1))
190 IFD=10THENC=C-1
200 K=K(ASC(MID$(L$,5,1))-65)
210 Y=41+B-C/8-K/48
220 RETURN
230 PRINT:PRINT"TOTAL:"T:END

READY.
```

El programa está preparado para trabajar entre los márgenes —NN01H y +MM80D, que se pueden alterar modificando las líneas 30 y 40. Asimismo cuando se le da un QTH Locator XXnnl (que no existe) lo toma como XXnnA.

Esperamos facilitaros la tarea de composición de «logs» a los que tengáis un VIC-20 como componente de la estación. ■

SUPEREXPANDER

dos programas de demostración

por **PERE MASATS**

Estos dos programas cuyos listados se dan en la figura están pensados para demostrar las posibilidades del cartucho SUPEREXPANDER.

Dado que hemos detectado algunas dificultades por parte de los usuarios a la hora de empezar a trabajar con los gráficos, creemos oportuno aconsejar que se introduzca en memoria el primero de ellos para ver sobre la marcha cómo trabaja cada instrucción. Si en un momento dado cambiamos algún valor que afecte a los comandos de gráficos podremos ver su efecto mucho mejor que escribiendo un largo (y aburrido) artículo sobre el tema. En lo que respecta a este programa hay que hacer una advertencia: debido a una misteriosa mala pasada de BUG, en la impresora no aparece el símbolo perteneciente a «F inversa» que sale en el listado de pantalla cuando en el SUPEREXPANDER se selecciona el modo «sonido». Esto se obtiene en las instrucciones PRINT al pulsar las teclas CTRL y «flecha a la izquierda». Esto debe añadirse en las líneas 915, 920, 925, 930, 935, 940, 945, 950, 1010, 1055 y 2000.

El segundo de los programas realiza sobre la pantalla una serie interminable de dibujos que van cambiando sin repetirse nunca, de manera que uno puede quedarse embobado delante del televisor horas y horas (como si en TVE alguien se hubiera equivocado y se estuviera emitiendo un programa interesante). A destacar, el sistema de simetrías que da a los dibujos una estética muy sugestiva. ■

PRIMER PROGRAMA

```
10 REM DEMOSTRACION DEL SUPER EXPANDER
20 POKE36879,59
30 GOSUB1200
40 PRINT"[illegible]       GRAFICOS":GOSUB1300
50 FORG=1TO100:NEXT
60 REM SELECCIONAR GRAFICOS
70 GRAPHIC2
80 REM SELEC. COLORES
90 COLOR3,3,0,10:GOSUB1500:GOSUB2100
100 REM DIBUJA CUADRADO
110 CHAR18,7,"CUADRADO":GOSUB2500
120 DRAW2,400,450TO600,450:GOSUB2000
130 DRAW2TO600,650:GOSUB2000
140 DRAW2TO400,650:GOSUB2000
150 DRAW2TO400,450:GOSUB2000:SCNCLR:GOSU
B1500
160 REM ESCRIBIR TEXTO
170 REM CIRCULO
180 CHAR18,7,"CIRCULO":GOSUB2500
190 CIRCLE2,512,512,70,100:GOSUB2100:SCN
CLR:GOSUB1500
200 REM DIBUJA TRIANGULO
210 CHAR18,7,"TRIANGULO":GOSUB2500
220 DRAW2,550,450TO350,650:GOSUB2000
230 DRAW2TO750,650:GOSUB2000
240 DRAW2TO550,450:GOSUB2000:SCNCLR:GOSU
B1500
250 REM DIBUJA ELIPSE
260 CHAR18,8,"ELIPSE":GOSUB2500
270 CIRCLE2,512,512,200,100:GOSUB2100:SC
NCLR:GOSUB1500
280 REM DIBUJA UN ARCO
290 CHAR18,9,"ARCO":GOSUB2500
300 CIRCLE2,512,512,70,100,25,75:GOSUB21
00:SCNCLR:GOSUB1500
310 REM DIBUJA PUNTOS
320 CHAR18,8,"PUNTOS":GOSUB2500
330 POINT2,400,450,400,650,600,650,600,4
50:GOSUB2100:SCNCLR:GOSUB1500
340 CHAR18,3,"CIRCULO PINTADO":GOSUB2500
350 CIRCLE2,512,512,70,100:GOSUB2100:REG
ION7
360 REM PINTAR FORMAS
370 PAINT2,512,512:GOSUB2100:REGION0:SCN
CLR:GOSUB1500
380 REGION0:CHAR18,3,"CUADRADO PINTADO":
GOSUB2500
390 DRAW2,400,450TO600,450:GOSUB2100
400 DRAW2TO600,650:GOSUB2100
410 DRAW2TO400,650:GOSUB2100
420 DRAW2TO400,450:GOSUB2100
430 REGION2:PAINT2,512,512:GOSUB2100:SCN
CLR:GOSUB1500
440 REGION0:CHAR18,3,"ELIPSE PINTADA":GO
SUB2500
450 CIRCLE2,512,512,200,100:GOSUB2100
460 REGION6:PAINT2,512,512:GOSUB2100:SCN
CLR:GOSUB1500
470 REGION0:CHAR18,3,"TRIANGULO PINTADO"
:GOSUB2500
480 DRAW2,550,450TO350,650:GOSUB2100
490 DRAW2TO750,650:GOSUB2100
500 DRAW2TO550,450:GOSUB2100
510 REGION5:PAINT2,512,512:GOSUB2100:REG
ION0:GOSUB1500
530 GRAPHIC0:GOSUB1200
540 PRINT"[illegible]          MUSICA":GOSUB1300:
GRAPHIC2:GOSUB3000
560 SOUND225,225,225,0,9:FORG=1TO2000:NE
XT:SOUND0,0,0,0,0:GOSUB3000
570 R=400
580 REM DIBUJAR PENTAGRAMA
590 DRAW2,1,RTO999,R:GOSUB2100
600 R=R+100:IFR>800THEN630
610 GOTO590
620 REM DIBUJAR CLAVE
630 CIRCLE2,200,900,7,14
640 DRAW2,200,900TO200,325
650 CIRCLE2,200,400,52,75,75,25
660 CIRCLE2,200,663,130,196,25,75
670 CIRCLE2,200,700,100,147,75,25
680 CIRCLE2,200,625,60,80,5,75
690 CIRCLE2,250,660,7,14
700 GOSUB2500
710 REM DIBUJAR NOTAS
720 R=300:R1=900:A=0
730 REGION0
740 CIRCLE2,R,R1,28,40
750 DRAW2,R+28,R1TOR+28,R1-200
760 A=A+1:ONAGOSUB915,920,925,930,935,94
0,945,950
770 R=R+90:R1=R1-50
780 IFR>1000THENGOSUB1000
790 GOTO740
800 FORZ1=1TO50000:NEXT
855 REM TOCAR NOTAS
915 PRINT"V9T6S202C":RETURN
920 PRINT"V9T6S202D":RETURN
925 PRINT"V9T6S202E":RETURN
930 PRINT"V9T6S202F":RETURN
935 PRINT"V9T6S202G":RETURN
940 PRINT"V9T6S202A":RETURN
945 PRINT"V9T6S202B":RETURN
950 PRINT"V9T6S203T7C":RETURN
```

(pasa a la pág. siguiente)

PRIMER PROGRAMA

(continuación)

```
1000 FORG=1TO500:NEXT
1005 REM VER NOTAS
1010 AA$="V9T4S2O2"
1015 REGION6
1020 CHAR4,10,"DO  ":PRINTAA$"CR";
1025 CHAR4,10,"RE  ":PRINT"DR";
1030 CHAR4,10,"MI  ":PRINT"ER";
1035 CHAR4,10,"FA  ":PRINT"FR";
1040 CHAR4,10,"SOL":PRINT"GR";
1045 CHAR4,10,"LA  ":PRINT"AR";
1050 CHAR4,10,"SI  ":PRINT"BR";
1055 CHAR4,10,"DO":PRINT"T8O3C"
1060 FORG=1TO2000:NEXT:SCNCLR
1065 CHAR2,2,"PULSE S"
1070 CHAR4,2,"PARA REPETIR"
1075 CHAR6,2,"N PARA TERMINAR"
1085 GETS$:IFS$<>"S"ANDS$<>"N"THEN1085
1090 IFS$="S"THEN1100
1095 GRAPHIC0:END
1100 GRAPHIC0:FORG=1TO100:NEXT:GOTO10
1195 REM CAMBIAR PANTALLA
1200 POKE36879,152:RETURN
1295 REM RODAR PANTALLA
1300 X1=36865:FORJ=150TO35STEP-1:POKEX1,
J:FORG=1TO10:NEXTG,J:RETURN
1499 FORG=1TO20000:NEXT
1500 CHAR2,6,"GRAFICOS":GOSUB2500:RETURN
2000 PRINT"V9T2S3O3C":FORS=1TO500:NEXT:R
ETURN
2100 FORQQ=1TO500:NEXT:RETURN
2500 FORW=1TO1000:NEXT:RETURN
3000 CHAR2,7,"MUSICA":RETURN

READY.
```

SEGUNDO PROGRAMA

```
100 REM PROGRAMA PARA REALIZAR DIBUJOS A
LEATORIOS
110 DIMX(4,2,8),Y(4,2,8)
120 FORI=1TO4000:NEXT
130 GOSUB400:B=RR
140 GOSUB400:Z=RR:IFZ=BTHEN140
150 R=INT(RND(1)*14)+8
160 P=INT(RND(1)*6)+2
170 FORI=1TO2:FORNA=1TOP
180 X(1,I,NA)=INT(RND(1)*600)+10
190 Y(1,I,NA)=INT(RND(1)*600)+10
200 X(2,I,NA)=1020-X(1,I,NA)
210 Y(2,I,NA)=Y(1,I,NA):X(3,I,NA)=X(1,I,
NA):Y(3,I,NA)=1020-Y(1,I,NA)
220 X(4,I,NA)=1020-X(1,I,NA)
230 Y(4,I,NA)=1020-Y(1,I,NA)
240 NEXT:NEXT
250 NA=NA-1
260 FORI=1TOR:FORF=0TOI-1:SP=SP+1/R:NEXT
F
270 FORD=1TO4
280 FORJ=1TONA
290 XX(J)=X(D,1,J)+(SP*(X(D,2,J)-X(D,1,J
))):YY(J)=Y(D,1,J)+(SP*(Y(D,2,J)-Y(D,1,J
)))
300 IFJ>=2THENGOSUB340
310 NEXTJ:NEXTD:SP=0:NEXTI
320 FORII=1TO4000:NEXT:SCNCLR:GOTO130
330 SCNCLR:GOTO130
340 REM DIBUJAR
350 GRAPHIC2
360 COLORB,B,Z,Z
370 DRAW2,X(D,1,1),Y(D,1,1)TOX(D,1,2),Y(
D,1,2)
380 DRAW2,XX(J-1),YY(J-1)TOXX(J),YY(J)
390 RETURN
400 RR=INT(RND(1)*8):RETURN

READY.
```

MINI-NOTAS

algo sobre INT

En álgebra la parte entera de un número se define como el entero inmediatamente inferior. Así:

PRINT INT (1.3)

1

es correcto, pero:

PRINT INT (—1.3)

—2

ya no lo es tanto, pues lo que cabría esperar como resultado es "—1". Si se desea el próximo entero **inferior,** la instrucción debe ser:

PRINT INT (ABS (—1.3)) * SGN (—1.3)

Por descontado "—1.3" será seguramente una variable de su programa. Una última nota: Las variables enteras realizan automáticamente una operación INT con las variables en punto flotante (las normales).

A% = INT (A)

Es lo mismo que:

A% = A

■

MINI-NOTAS

la instrucción IF... THEN... ELSE

por **PERE MASATS**

Esta no es una instrucción standard de BASIC. Esto quiere decir que no está en el conjunto de instrucciones que se consideran las mínimas imprescindibles para que un intérprete pueda llamarse DE BASIC y, por lo tanto, no viene incorporada en los intérpretes de COMMODORE. Sin embargo, ciertas opciones ROM para los sistemas 4000 y 8000 sí permiten utilizarla. También hay que decir que tanto el VIC como los sistemas 4000 y 8000 trabajan con instrucciones que no son standard de BASIC como SYS y otras.

Si bien no podemos trabajar directamente con IF...THEN...ELSE, vamos a intentar sustituirla por alguna secuencia de BASIC ortodoxo. Para ello veamos qué hace esta instrucción. Las dos primeras partes de la misma son iguales que las que conocemos y la diferencia está solamente en la tercera el ELSE que aquí podríamos traducir por SINO. Así, si la condición señalada por IF (si condicional) se cumple, se ejecuta lo que indica THEN (entonces) pero, si no se cumple, lo que se ejecuta es la parte ELSE (sino).

Así, si tenemos:

100 IF X=0 THEN 120 ELSE X=X+1

Y lo traducimos por:

100 IF X=0 THEN 120 : X=X+1

esto nunca funcionará porque, si la condición se cumple, la próxima línea que se ejecutará será la 120 y, si no se cumple, el programa pasará a la línea siguiente, la 110, en el caso más probable. De esto podemos deducir cuál será la solución a nuestro problema:

100 IF X=0 THEN 120
105 X=X+1

Para poder incorporarlo todo en una sola línea:

100 ON —(X=0) GOTO 120 : X=X+1

Las pruebas CIERTO/FALSO en el BASIC de COMMODORE dan como resultado un «0» si es FALSO y un «—1» si es cierto (pruebe: PRINT 4=5 y PRINT 5=5). Un número negativo en una instrucción ON...GOTO daría ?ILLEGAL QUANTITY ERROR. Por ello cambiamos el signo con —(X=0). Afortunadamente en esta instrucción un número fuera de rango (en nuestro caso «0») no obliga a la ejecución de la siguiente línea sino que continúa ejecutando el resto de la línea actual. Esto es cierto también si no hay más números de línea a continuación del GOTO. ■

programa de utilidad para la impresora

por **P. MASATS**

```
READY.

10 OPEN12,4:SI$=CHR$(15):BS$=CHR$(8)
20 PRINT#12,SI$"┌──────────┬──────────┐"BS$
30 PRINT#12,SI$"| ESTO ES  UNA        |"BS$
40 PRINT#12,SI$"|          |          |"BS$
50 PRINT#12,SI$"|          |          |"BS$
60 PRINT#12,SI$"| DEMOSTRACION        |"BS$
70 PRINT#12,SI$"└──────────┴──────────┘"BS$
80 PRINT#12,SI$:CLOSE12

READY.

┌─────────────┐
| ESTO ES UNA |
|             |
| DEMOSTRACION|
└─────────────┘

READY.
```

Fig. 1

N. Tallada, de Barcelona, nos manda un programa que permite resolver un viejo problema de la impresora del VIC. Consiste en que si queremos hacer una impresión utilizando los caracteres gráficos del teclado, la separación entre líneas nos impide obtener la unión de los dibujos de una línea con la siguiente. La solución parece que consiste en definir el modo gráfico antes de hacer la impresión, pero entonces necesitamos definir los caracteres gráficos con lo que perdemos las posibilidades del teclado. El truco que listamos en la figura 1 y del que damos una muestra más abajo consiste en definir el modo gráfico sólo durante la operación de avance de línea mediante las variables SI$ (paso a modo alfanumérico) y BS$ (paso a modo gráfico). ■

Para cartucho 3K

JUEGOS

dos interesantes juegos

por **P. MASATS**

Sin que sirva de precedente y aprovechando las festividades de principio de año incluimos en este número otros dos juegos

El primero de ellos tiene un cierto aspecto educativo, pues consiste en dar el nombre en letras de los números que van saliendo en pantalla. Este programa nos ha sido remitido por nuestro compañero Antonio González, de Microelectrónica-Madrid, que lo escribió para su hija.

El segundo es uno más de los denominados «de marcianitos» y como en CLUB COMMODORE ha de haber de todo...

JUEGO NÚM. 1

```
10 REM ************************
20 REM *                      *
30 REM *  EL JUEGO DE LOS     *
40 REM *      NUMEROS.        *
50 REM * MICROELECTRONICA Y   *
60 REM * CONTROL; BARCELONA   *
70 REM *                      *
80 REM ************************
90 INPUT"DIME TU NOMBRE";NO$
100 DIMTE$(54)
110 BL$=" ":FORX=1TO6:BL$=BL$+BL$:NEXTX
120 DATA"CIEN-TO ","DOS-CIEN-TOS ","TRES
-CIEN-TAS ","CUA-TRO-CIEN-TOS "
130 DATA"QUI-NIEN-TOS "
140 DATA"SEIS-CIEN-TOS ","SE-TE-CIEN-TOS
 ","OCHO-CIEN-TOS ","NO-VE-CIEN-TOS "
150 DATA"TREIN-TA ","CUA-REN-TA ","CIN-C
UEN-TA ","SE-SEN-TA ","SE-TEN-TA "
160 DATA"OCHEN-TA ","NO-VEN-TA "
170 DATA"Y UNO ","Y DOS ","Y TRES ","Y C
UATRO ","Y CIN-CO ","Y SEIS "
180 DATA"Y SIETE ","Y OCHO ","Y NUEVE ",
"UNO ","DOS ","TRES ","CUA-TRO "
190 DATA"CINCO ","SEIS ","SIETE ","OCHO
","NUEVE ","DIEZ ","ONCE ","DOCE "
200 DATA"TRECE ","CA-TOR-CE ","QUIN-CE "
,"DIECI-SEIS ","DIECI-SIETE "
210 DATA"DIECI-OCHO ","DIECI-NUE-VE ","V
EINTE ","VEIN-TI-UNA ","VEIN-TI-DOS "
220 DATA"VEIN-TI-TRES ","VEIN-TI-CUA-TRO
 ","VEIN-TI-CINCO ","VEIN-TI-SEIS "
230 DATA"VEIN-TI-SIETE ","VEIN-TI-OCHO "
,"VEIN-TI-NUEVE "
240 FORX=1TO54:READTE$(X):NEXTX
250 A=INT(100*RND(0)):IK$=STR$(A):LI$=""
:PRINT"■QUE NUMERO ES ESTE"A
260 INPUTA$:GOSUB340
270 B$="":FORX=1TOLEN(LI$)
280 IFMID$(LI$,X,1)<>"-"THENB$=B$+MID$(L
I$,X,1)
290 NEXT:IFB$<>A$+" "THENPRINT"■MUY MAL
";NO$;", EL NUMERO ES:":PRINT"■■"B$:GOTO
310
300 PRINT"■MUY BIEN ";NO$
310 PRINT"■■■ PULSA UNA TECLA PARA
 CONTINUAR           ■"
320 GETA$:IFA$=""THEN320
330 GOTO250
340 IK$=RIGHT$(BL$+IK$,9)
350 FORI=1TO3:N$(I)=MID$(IK$,3*I-2,3):C(
I+4)=VAL(N$(I)):NEXT
360 FORI=7TO9:N$(I)="":NEXT:IFVAL(IK$)=0
THENN$(9)="CERO ":GOTO450
370 IFC(5)=0THEN400
380 IFC(5)=1THENN$(7)="UN MILLON ":GOTO4
00
390 I=1:GOSUB460:GOSUB580:N$(7)=LI$+"MIL
LONES "
400 IFC(6)=0THEN430
410 IFC(6)=1THENN$(8)="\¬L ":GOTO430
420 I=2:GOSUB460:GOSUB580:N$(8)=LI$+"MIL
 "
430 IFC(7)=0THEN450
440 I=3:GOSUB460:GOSUB580:N$(9)=LI$
450 LI$="":FORX=7TO10:LI$=LI$+N$(X):NEXT
:RETURN
460 LI$="":FORJ=1TO3
470 C(J+7)=VAL(MID$(N$(I),J,1)):NEXT
480 IFC(8)<>0THENLI$=TE$(C(8))
490 LJ=C(9)*10+C(10):IFLJ=0THEN570
500 IFLJ>=30THEN520
510 LI$=LI$+TE$(25+LJ):GOTO570
520 IFC(8)=0THEN540
530 IFC(9)=0THEN550
540 LI$=LI$+TE$(7+C(9))
550 IFC(10)=0THEN570
560 LI$=LI$+TE$(16+C(10))
570 RETURN
580 IFLI$="CIEN-TO "THENLI$="CIEN "
590 RETURN

READY.
```

JUEGO NÚM. 2

```
1 DATA191,2,191,2,191,2,207,8,223,8,0,4,
219,2,217,2,213,2,231,8,223,8
2 DATA0,4,219,2,217,2,213,2,231,8,223,8,
219,2,217,2,219,2,213,12,0,4
5 DATA16,16,16,16,16,16,16,16,0,0,0,255,
0,0,0,0
6 DATA64,0,1,16,0,64,1,130,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0
7 DATA68,108,124,40,56,16,16,0,8,12,14,2
8,252,122,38,1
8 DATA0,224,120,46,120,224,0,0,1,38,122,
252,28,14,12,8
9 DATA0,16,16,56,40,124,108,68,128,100,9
5,63,56,112,48,16
10 DATA0,7,30,116,30,7,0,0,16,48,112,56,
63,95,100,128
11 DATA1,2,4,8,16,32,64,128,128,64,32,16
,8,4,2,1
13 DATA60,60,126,65,255,255,60,60,12,28,
30,127,255,255,252,56
15 DATA0,8,24,56,48,32,0,0,0,0,56,252,25
5,255,127
16 POKE52,28:POKE56,28:CLR:POKE36878,15
17 POKE36879,25:PRINT"■■■■■■■■*ROCKS*"
18 PRINT"■■■■  << CONTROLS>>"
19 PRINT"■ 1 AND 2 TURNS SHIP":PRINT"■ 8
 AND 9 MOVES SHIP":PRINT"■ 0 FIRES A BEA
M"
20 PRINT"■■   TO JUMP HYPERSPACE    USE
THE SPACE BAR"
21 FORZ=1TO22:READA:POKE36876,A:POKE3687
5,A
22 READB:FORX=1TOB*100:NEXT:POKE36876,0:
POKE36875,0:NEXTZ
23 PRINT"■■PRESS ANY KEY TO START"
24 GETA$:IFA$=""THEN24
30 FORA=232TO382:READB:POKE5120+A,B:NEXT
35 FORA=382TO470:POKE5120+A,PEEK(32768+A
):NEXT
40 POKE36869,253:POKE36866,PEEK(36866)OR
128
50 SC=0:X2=34
60 X=7901:X1=5
65 POKE0,32
70 PRINT"■":POKE36879,8
80 FORA=7658TO7679:POKEA,33:NEXT:FORA=81
64TO8186:POKEA,33:NEXT
90 FORA=7680TO8164STEP22:POKEA,33:NEXT
100 FORA=1TO4
110 IFPEEK(R(A))>33THEN140
120 IFA=1THENR(1)=INT(20*RND(1)+7703):T(
1)=INT(3*RND(1)+45):M(1)=INT(3*RND(1)+21
)
121 IFA=1ANDSC>300ANDINT(5*RND(1))=2THEN
T(1)=44:R(1)=7703+X1:M(1)=22
124 IFA=2THENR=INT(20*RND(1)):R(2)=7703+
(R*22):T(2)=INT(3*RND(1)+45):M=INT(6*RND
(1))
125 IFA=2ANDM<1THENM(2)=23
126 IFA=2ANDM>1THENM(2)=-23
127 IFA=2ANDSC>300ANDINT(5*RND(1))=2THEN
T(2)=44:R(2)=X:M(2)=1
129 IFA=3THENR=INT(20*RND(1)):R(3)=7723+
(R*22):T(3)=INT(3*RND(1)+45):M=INT(4*RND
(1))
130 IFA=3THENM(3)=21
131 IFA=3ANDM=2THENM(3)=21
132 IFA=3ANDM=1THENM(3)=-1
133 IFA=3ANDSC>300ANDINT(5*RND(1))=2THEN
T(3)=44:R(3)=X+20:M(3)=-1
135 IFA=4THENR(4)=INT(20*RND(1)+8143):T(
4)=INT(3*RND(1)+45):M(4)=INT(4*RND(1)-23
)
136 IFA=4ANDSC>300ANDINT(5*RND(1))=2THEN
T(4)=44:R(4)=8143+X1:M(4)=-22
140 POKER(A),32:R(A)=R(A)+M(A)
145 IFT(A)=44THENPOKE36876,200
150 IFR(A)=X+X1THEN600
155 POKE36876,0
160 IFPEEK(R(A))<>32THEN180
170 POKER(A),T(A):POKER(A)+30720,A+2
180 J=PEEK(197)
181 IFJ=0THENX2=X2+1:IFX2=42THENX2=34
182 IFJ=56THENX2=X2-1:IFX2=33THENX2=41
183 IFJ=4ANDX1<15THENPOKEX+X1,32:X1=X1+1
184 IFJ=32THENPOKEX+X1,32:X=7681+(22*(IN
T(RND(1)*20))):X1=INT(20*RND(1))
185 IFJ=32THENSC=SC-100:IFSC<0THENSC=0
186 IFJ=60THEN400
187 PRINT"■■■■■■■■■"SC
190 IFJ=59ANDX1>5THENPOKEX+X1,32:X1=X1-1
200 POKEX+X1,X2:POKEX+X1+30720,7
300 NEXT:GOTO100
400 IFX2=34THENF=22:F1=29
401 IFX2=35THENF=23:F1=43
402 IFX2=36THENF=1:F1=30
403 IFX2=37THENF=-21:F1=42
404 IFX2=38THENF=-22:F1=29
405 IFX2=39THENF=-23:F1=43
406 IFX2=40THENF=-1:F1=30
407 IFX2=41THENF=21:F1=42
410 F2=X+X1:J=250
420 F2=F2+F:IFPEEK(F2)<>32THEN450
430 POKEF2,F1:POKE36877,J:J=J-2:POKEF2+3
0720,2
440 GOTO420
450 U=PEEK(F2)
455 IFPEEK(F2)<34ORPEEK(F2)>47THEN510
460 POKEF2,31
470 POKE36875,150:POKE36877,150
475 IFU=44THENSC=SC+50:U=32:GOTO490
480 SC=SC+10:U=32
490 PRINT"■■■■■■■■■"SC
510 FORW=X+X1+FTOF2STEPF:POKEW,32:NEXT
515 POKE36877,0:POKE36875,0:POKEF2,U
520 GOTO200
600 FORX=1TO1000:NEXT:POKE36869,240:POKE
36866,150:POKE36879,25:PRINT"■■":POKE368
78,0:END

READY.
```

■

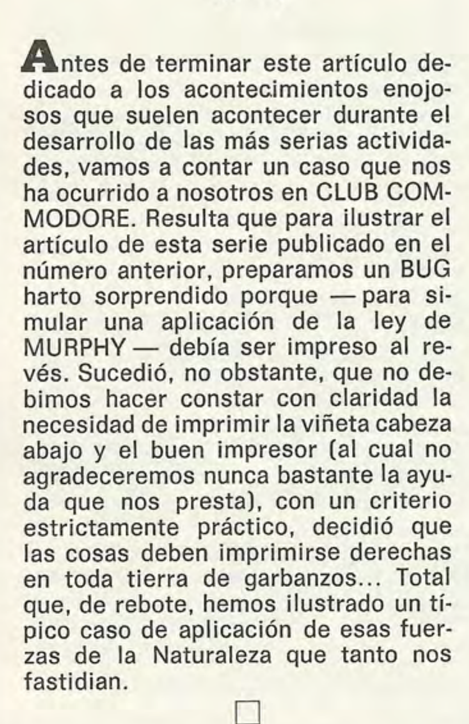
FILOSOFÍA APLICADA



la ley de Murphy y sus corolarios

(y II)

por **PERE MASATS**



Antes de terminar este artículo dedicado a los acontecimientos enojosos que suelen acontecer durante el desarrollo de las más serias actividades, vamos a contar un caso que nos ha ocurrido a nosotros en CLUB COMMODORE. Resulta que para ilustrar el artículo de esta serie publicado en el número anterior, preparamos un BUG harto sorprendido porque —para simular una aplicación de la ley de MURPHY— debía ser impreso al revés. Sucedió, no obstante, que no debimos hacer constar con claridad la necesidad de imprimir la viñeta cabeza abajo y el buen impresor (al cual no agradeceremos nunca bastante la ayuda que nos presta), con un criterio estrictamente práctico, decidió que las cosas deben imprimirse derechas en toda tierra de garbanzos... Total que, de rebote, hemos ilustrado un típico caso de aplicación de esas fuerzas de la Naturaleza que tanto nos fastidian.

□

Un capítulo especial dedicado a los electrónicos:

—Todo cable cortado a la medida exacta resultará corto.

—Las tolerancias se acumularán en el mismo sentido para complicar la operación de montaje.

—Dos equipos idénticos controlados en condiciones idénticas no serán nunca idénticos a la hora de utilizarlos.

—La disponibilidad de un componente será inversamente proporcional a su necesidad.

—Si se necesitan N componentes para la realización de un proyecto habrá N-1 en stock.

—Si se necesita una resistencia de un determinado valor, este valor no existirá y no se podrá obtener por ninguna combinación de valores standard en serie o en paralelo.

—Una herramienta que se nos escape de las manos caerá o en el lugar más inaccesible o sobre el componente más frágil. (También conocida como LEY DE LA GRAVEDAD SELECTIVA.)

—Todo dispositivo tomado al azar de un grupo con una fiabilidad de un 99 % formará parte siempre del conjunto del 1 %.

—Cada vez que se conecte una línea trifásica, el orden de las fases estará invertido.

—Un motor girará siempre en sentido contrario al deseado.

—La probabilidad de ausencia de una cota en un plano está en proporción directa a su importancia.

—Los componentes intercambiables no lo serán jamás.

—La probabilidad de fallo de un componente, de un conjunto, de un subsistema o de un sistema es inversamente proporcional a su facilidad de reparación o sustitución.

—Si un prototipo funciona correctamente dejará de hacerlo en cuanto se inicie la producción.

—Los componentes que no deben ni pueden ser nunca montados incorrectamente lo serán en producción.

—Cuando se conecta un «tester» para medir corriente continua siempre está a máxima sensibilidad y con la polaridad invertida.

—Siempre se caen los componentes más delicados.

—Los registradores gráficos depositan más tinta sobre los seres humanos que sobre el papel.

—Un circuito protegido contra todo fallo será el primero en estropearse. Si no es así, provocará la destrucción de los demás.

—Un disyuntor de protección instantánea de la alimentación se disparará siempre demasiado tarde.

—Un transistor protegido por un fusible rápido protegerá al fusible, quemándose el primero.

—Un oscilador con auto disparo se obstinará en no oscilar nunca espontáneamente.

—Un oscilador a cuarzo oscilará a una frecuencia distinta a la prevista (si oscila).

—Un transistor NPN será generalmente un PNP.

—Un condensador con un coeficiente de temperatura negativo empleado en un circuito crítico resultará tener un C.T. de 750 ppm/grado C.

—Una avería no se manifestará nunca antes de haber pasado la inspección final.

—Un componente o un instrumento corresponderá a las especificaciones anunciadas sólo durante el tiempo necesario para pasar la inspección de entrada.

—Si se reemplaza un componente manifiestamente defectuoso en un instrumento que presenta una anomalía intermitente, ésta reaparecerá justo cuando se ponga en servicio.

—Después de fijar los 16 tornillos de una cubierta de un instrumento, es cuando nos damos cuenta que hemos olvidado una conexión vital y además caemos en la cuenta que hemos confundido las cubiertas.

—Cuando un instrumento está completamente montado es normal que sobren componentes en la mesa de trabajo.

—Las juntas herméticas pierden.

TRECE AXIOMAS BÁSICOS DE LA CIENCIA

1. Teorema de Patrick: Si un experimento funciona es probable que sea porque utilizamos equipo inadecuado.

2. Constante de Shinner: Es la cantidad que, al multiplicarla, dividirla, sumarla o restarla del resultado obtenido, nos da el resultado previsto.

3. Postulado de Horner: La experiencia varía proporcionalmente a la cantidad de equipo estropeado.

4. Ley de perversidad de los objetos inanimados: Todo objeto inanimado, independientemente de su composición y configuración, puede producir en cualquier momento, de forma totalmente inesperada y por razones que permanecerán siempre oscuras y misteriosas, actos perversos en contra de nuestros deseos y proyectos.

5. Axioma de Allen: Si todo falla, léase el manual de instrucciones.

6. Corolario de la compensación: Se puede considerar un éxito todo experimento que nos proporcione un 50 % de resultados equivocados, con referencia a la teoría propuesta.

7. Ley de Gumperson: La probabilidad de que se produzca un hecho determinado, es inversamente proporcional al deseo que tenemos de que se produzca.

8. Principio general de la Ciencia: Por definición, cuando investigamos lo desconocido, no sabemos lo que vamos a encontrar.

9. Regla de Kettering: Si algo no funciona es por una razón diferente a la que creemos.

10. Ley de Gummidge: La experiencia de un investigador científico, varía inversamente con el número de palabras que el público en general entienda de tal sujeto.

11. Principio de utilidad: Ningún experimento es un fracaso total; como mínimo, sirve como un mal ejemplo.

12. Ley de Anderson: Nunca se estropea el componente del cual tenemos recambio.

13. Corolario final: Cuando suene el despertador por la mañana, páralo, pero no te levantes pues es posible que todo te salga mal. (Alguien me dice que éste es un antiguo proverbio chino.)

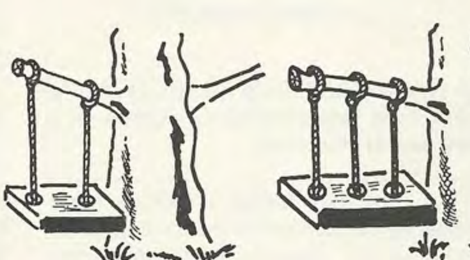
Como puede verse hemos reunido una lista exhaustiva de principios de indiscutible utilidad. No obstante, si algún lector cree que existe alguna otra «LEY BÁSICA DE LA NATURALEZA» que falte en esta relación queda invitado desde aquí a remitírnosla para — en caso de que no se pierda — proceder a publicarla.

□

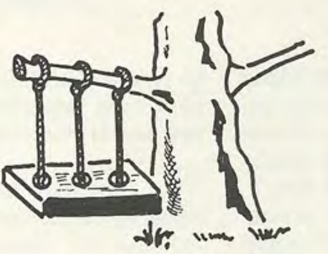
Y para terminar ofrecemos una pequeña ilustración de lo que — se dice que — suele ocurrir en las empresas modernas entre el momento en que un cliente decide comprar un artículo y lo que se le sirve en realidad. (Como es evidente se trata de una exageración representativa.) ■



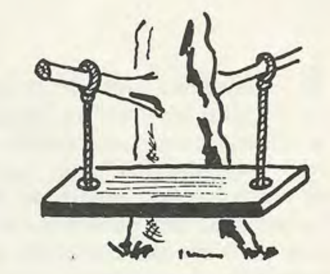
Lo que ofrece la publicidad



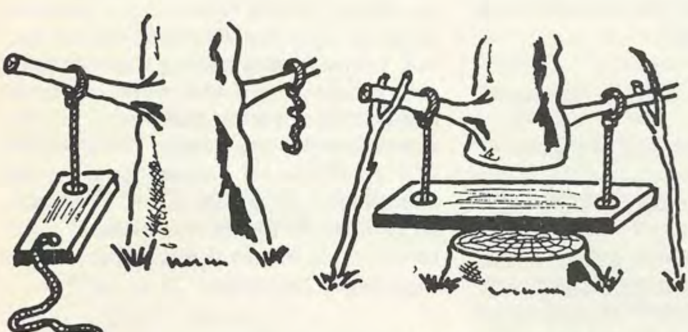
Lo que el cliente pide



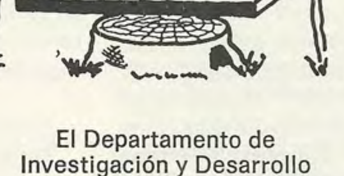
El pedido que cursa el vendedor



Lo que el jefe de producción entiende



Lo que marketing opina



El Departamento de Investigación y Desarrollo resuelve el problema



Lo que el jefe comercial impone

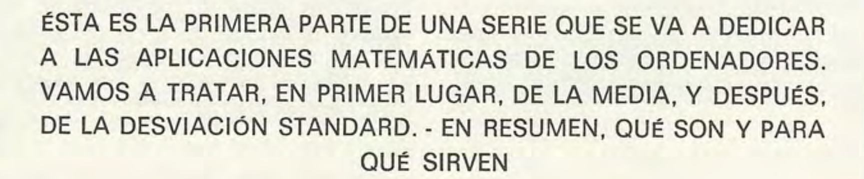


El producto final

NUMEROLOGÍA NO ESOTÉRICA

introducción a la estadística: la media y la desviación standard




ÉSTA ES LA PRIMERA PARTE DE UNA SERIE QUE SE VA A DEDICAR A LAS APLICACIONES MATEMÁTICAS DE LOS ORDENADORES. VAMOS A TRATAR, EN PRIMER LUGAR, DE LA MEDIA, Y DESPUÉS, DE LA DESVIACIÓN STANDARD. - EN RESUMEN, QUÉ SON Y PARA QUÉ SIRVEN

por **PERE MASATS**

Si deseamos evaluar la velocidad de un automóvil en un recorrido determinado para poder compararla con otro vehículo, decimos: este coche ha realizado una media de 90 Km/h. Es decir, lo que generalmente utilizamos es **la media.** Tomamos el tiempo invertido y lo dividimos por la distancia recorrida. Así tenemos una velocidad que es la que el automóvil hubiera llevado si todo el tiempo hubiera circulado a la misma velocidad. Con esto obtenemos dos cosas:

1. Tenemos una cantidad representativa del comportamiento del automóvil y de su conductor y, en cierta medida, independiente de factores externos como el viento pues, en estadística, se supone que, a la larga, estos factores se anulan a sí mismos. Para aclarar esto último téngase en cuenta que el viento que sopla de frente en un momento dado, al cabo de una o dos curvas, sopla por detrás o que la pérdida de velocidad debida a una cuesta queda compensada por la aceleración en la siguiente pendiente. Debido a esto, la media será **más representativa cuanto mayor sea el recorrido.**

2. Con esta media podemos comparar el comportamiento de dos automóviles en recorridos de diferente longitud pues la media es relativamente independiente del kilometraje que se haya realizado. Teóricamente es así pero, en la práctica, todos sabemos que el rendimiento del conductor no es el mismo durante la primera hora de conducción que cuando lleva ocho horas al volante.

La fórmula para obtener la media es:

$$\text{MEDIA} = \frac{\text{SUMA DEL TIEMPO}}{\text{SUMA DEL RECORRIDO}}$$

En la fórmula hemos escrito SUMA DEL TIEMPO y SUMA DEL RECORRIDO en vez de TIEMPO Y RECORRIDO por la siguiente razón que exponemos a continuación. Imaginemos que queremos saber la media del peso de un conjunto de latas de sardinas. Entonces la fórmula será:

$$\text{MEDIA} = \frac{\text{SUMA DEL PESO DE LAS LATAS}}{\text{NÚMERO DE LATAS}}$$

Utilizando esta fórmula, la media dará un valor que nos dirá qué peso tendría una lata ideal fabricada con el mismo sistema que las anteriores o, dicho de otra forma, el peso probable que tendrá la próxima lata que saquemos del montón del que hemos sacado las anteriores. Así tenemos la primera posibilidad de «predecir el futuro» que caracteriza a la estadística que es la principal razón por la cual los políticos profesionales la utilizan tanto. (Un inciso jocoso: el general De Gaulle decía que las estadísticas son como los bikinis: que enseñan mucho pero ocultan lo esencial).

LA DESVIACIÓN STANDARD

Como hemos visto, la media nos da una idea de qué tipo de valor podemos encontrar cuando examinamos un conjunto de artículos (también se les llama «muestras») pero supongamos que tenemos dos conjuntos, de tres botellas de refresco cada uno, que en la etiqueta llevan escrito: «CONTENIDO: 1 LITRO». Al primer conjunto, le llamaremos «MARCA A» y, al segundo, «MARCA B». Si medimos el contenido de la marca A encontramos los siguientes valores:

Botella 1: 950 cl.
Botella 2: 1.000 cl.
Botella 3: 1.050 cl.

Como se ve, la media es: (950 + 1.000 + 1.050)/3 = 1.000 cl. o sea 1 litro. En cambio, en la marca B tenemos:

Botella 1: 900 cl.
Botella 2: 1.000 cl.
Botella 3: 1.100 cl.

Con lo que la media es: (900 + 1.000 + 1.100)/3 = 1.000 cl. que es la misma media que en la marca A pero si observamos los dos juegos de valores vemos que la media no nos dice que la marca B va más a su aire a la hora de embotellar sus productos. Pues bien, la desviación standard es lo que nos expresa la dispersión de un conjunto de muestras (para más claridad diremos que también se le llama DESVIACIÓN TÍPICA). Su fórmula es:

DESVIACIÓN STANDARD o [sigma] = SQR ((SUMA DE LAS DIFERENCIAS DE CADA VALOR Y LA MEDIA) ∧ 2/EL NÚMERO DE VALORES MENOS 1).

Esta cantidad representará la variación de cada una de las muestras referidas a un número dado que, en este caso, será la media. La desviación standard es tal que, por razones cuya demostración escapa al propósito de este artículo (y a la paciencia de muchos lectores), si tomamos el valor de la media y le sumamos [sigma] y se lo restamos obtendremos dos valores que serán los límites entre los que estará el 68 % de las muestras que tomemos. Si hacemos lo mismo pero con [sigma]*2 tenemos los valores que contienen el 95 % de la muestra y para [sigma]*3, el 99,7 %. Esto es así si el conjunto de valores que tomamos responden a lo que se llama una distribución normal que es a la que responden, generalmente, los fenómenos de la Naturaleza.

Para ver esto de una manera práctica podemos cargar y ejecutar el programa de la figura 1. Si entramos los datos de las dos marcas de refrescos obtendremos los siguientes valores:

MARCA A — MEDIA = 1.000 cl., DESVIACIÓN = 50 cl.

MARCA B — MEDIA = 1.000 cl., DESVIACIÓN = 100 cl.

```
10 REM PROGRAMA PARA CALCULAR LA MEDIA
20 REM Y LA DESVIACION ESTANDAR
30 REM MICROELECTRONICA Y CONTROL
40 REM P. MASATS
50 REM BARCELONA; 16-SEPT.-1982
60 M1=80
70 DIMX(80)
80 PRINT"###ESTADISTICA 1#####";
90 PRINT"CALCULO DE LA MEDIA Y ";
100 PRINT"LA DESVIACION ESTANDAR";
110 PRINT"++++++++++++++++++++++"
120 GOSUB220:REM ENTRAR DATOS
130 GOSUB310:REM CALCULAR MEDIA Y DESVIA
CION
140 PRINT
150 PRINT"PARA";N;"MUESTRAS"
160 PRINT"LA MEDIA ES:";M2
170 PRINT
180 PRINT"LA DESVIACION ES:"
190 PRINTS1
200 GETA$:IFA$=""THEN200
210 GOTO80
220 REM ENTRAR DATOS
230 INPUT"CUANTAS MUESTRAS";N
240 IFN>M1THEN230
250 IFN<2THEN420
260 FORI=1TON
270 PRINTI;
280 INPUTX(I)
290 NEXTI
300 RETURN
310 REM CALCULAR LA MEDIA Y LA DESVIACIO
N DE N MUESTRAS EN EL VECTOR X(I)
320 REM
330 S3=0
340 S4=0
350 FORI=1TON
360 S3=S3+X(I)
370 S4=S4+X(I)*X(I)
380 NEXTI
390 M2=S3/N
400 S1=SQR((S4-S3*S3/N)/(N-1))
410 RETURN
420 END

READY.
```

Fig. 1

Lo que nos da unas cifras que podemos asociar a las características de los datos que tenemos.

Vamos a examinar un ejemplo para ver la aplicación de estos porcentajes tan raros que resultan de sumarle y restarle a la media los múltiplos de [sigma]. Supongamos que tenemos que diseñar un puente de acero y — claro — hay que seleccionar la marca de acero que más nos convenga pero si sólo hacemos caso de la publicidad de los fabricantes vamos a acabar hechos un lío. Sin embargo, podemos pedir que se nos suministren para cada calidad de acero la media y la desviación standard de — por ejemplo — la resistencia. Supongamos que nos dan lo siguiente: Acero 1; 450 MPa. (megapascals) de media con una desviación típica de 10 MPa. Esto significa que un 99,7 % de las piezas construidas con este acero tendrán una resistencia entre 420 y 480 MPa. o sea que en el diseño hemos de calcular las dimensiones de las piezas teniendo en cuenta el peor caso: 420 MPa. Por otro lado, el Acero 2 nos da una media de 500 MPa. y una desviación de 40 MPa. ¿Cuál es mejor? Aparentemente el segundo pues su media es más alta pero si calculamos el significado de la desviación veremos que dentro de 3*[sigma] (99,7 %) podemos encontrar piezas de 500 +/— 40*3 = 380 a 620 MPa. que, utilizando el criterio del peor caso, nos obliga a prever la presencia de piezas de 380 MPa. con lo que el diseño queda seriamente afectado. ■

COLABORACIONES

resolución de sistemas de *n* ecuaciones y *N* incógnitas

por **A. CERDÓ**

Otra vez vamos con la sección despistada porque este programa es ideal para NUMEROLOGÍA NO ESOTÉRICA, pero como las colaboraciones hay que aprovecharlas cuando surgen y ésta es una aplicación muy frecuente en cálculo... De su autor no sabemos mucho, pero desde aquí le convocamos a participar con frecuencia dada la calidad de esta — su primera — colaboración.

Resolver un sistema de dos ecuaciones con dos incógnitas no presenta problema. Es una operación sencilla y rápida. Si el sistema es de tres ecuaciones la cosa se complica ligeramente pero, con cuidado y tiempo, se logran buenos resultados. Si el sistema es de 4, 5, 6... ecuaciones, ya el cálculo se hace muy laborioso y frecuentemente sujeto a errores.

El VIC-20 no permite el manejo **directo** de matrices, al carecer de las instrucciones MAT READ, MAT INPUT, etc., así como tampoco la multiplicación matricial directa ni el cálculo del determinante de una matriz.

El programa que he desarrollado permite la resolución de **n** ecuaciones con **N** incógnitas gracias a un subprograma (instrucciones 500 a 680) que calcula el valor de los determinantes.

La única precaución que hay que tomar es evitar que el primer coeficiente de la primera ecuación sea cero. Si así fuera, bastaría cambiar el orden de la primera ecuación y situarla en segundo lugar.

Para la resolución del sistema se ha seguido la regla de Kramer, que da los valores de las incógnitas dividiendo el determinante de cada una de ellas por el determinante del sistema.

El determinante del sistema contiene los coeficientes de las incógnitas. El determinante de cada una de las incógnitas se obtiene sustituyendo en el sistema los coeficientes de esa incógnita por los términos independientes.

Para resolver un determinante se han seguido los pasos siguientes:

1. Sumar a los elementos de cada columna, números proporcionales a los elementos de la primera columna, con el fin de que los elementos de la primera fila, excluyendo el (1,1), sean ceros. Con esta operación, el determinante no cambia de valor.
2. A continuación, desarrollar el determinante por el procedimiento de las menores.

Estos procesos se repiten una y otra vez hasta que la dimensión del determinante sea 2.

```
1 PRINT"□"
5 REM RESOLUCION DE SISTEMAS DE ECUACION
ES
10 PRINT"***********************"
20 PRINT"□SISTEMAS DE ECUACIONES"
30 PRINT"□***********************"
40 PRINT"ESTE PROGRAMA CALCULA"
50 PRINT"     LAS SOLUCIONES"
60 PRINT"    DE UN SISTEMA DE"
70 PRINT"       N ECUACIONES"
80 PRINT"           CON"
90 PRINT"       N INCOGNITAS"
100 PRINT"======================="
110 REM INTRODUCCION DE DATOS
120 INPUT"NO.DE INCOGNITAS";N:DIMAA(N,N)
,TT(N,N),T(N,N),L(N)
130 PRINT:PRINT"COEFICIENTES DE":PRINT
140 FORA=1TON
150 PRINT:PRINT"ECUACION "A
160 FORB=1TON
170 INPUTAA(A,B):TT(A,B)=AA(A,B)
180 NEXTB
190 NEXTA
200 PRINT:PRINT"TERM.INDEPENDIENTES"
210 FORA=1TON
220 INPUTL(A)
230 NEXTA
240 REM DETERMINANTE DEL SISTEMA
250 GOSUB500
260 D1=D
270 PRINT:PRINT"          □SOLUCIONES□"
280 FORR=1TON
290 FORA=1TON
300 FORB=1TON
310 TT(A,B)=AA(A,B)
320 NEXTB
330 NEXTA
340 REM DETERMINANTE DE LAS INCOGNITAS
350 FORQ=1TON
360 TT(Q,R)=L(Q)
370 NEXTQ
380 GOSUB500
390 DD=D/D1
400 PRINT:PRINT"□□□□"CHR$(64+R)"="DD
410 NEXTR
420 PRINT:PRINT"DESEA CONTINUAR(S/N)"
430 GETA$:IFA$=""THEN430
440 IFA$="N"THENEND
450 IFA$="S"THENCLR:PRINT:PRINT:GOTO120
460 GOTO430
470 END
500 REM RESOLUCION DE UN DETERMINANTE
510 FORZ=NTO2STEP-1
520 FORB=2TON
530 FORA=1TON
540 T(A,1)=TT(A,1):T(A,B)=TT(A,B)-TT(A,1
)*TT(1,B)/TT(1,1)
550 NEXTA
560 NEXTB
570 IFZ=2THEN670
580 FORA=1TON
590 T(A,2)=T(A,2)*T(1,1)
600 NEXTA
610 FORB=1TON-1
620 FORA=1TON-1
630 TT(A,B)=T(A+1,B+1)
640 NEXTA
650 NEXTB
660 NEXTZ
670 D=T(1,1)*T(2,2)-T(1,2)*T(2,1)
680 RETURN

READY.
```

(Fig. 1)

El programa dará mensaje de ¿DIVISION BY ZERO

ERROR IN 540

Cuando el elemento (1,1) = 0 en cuyo caso el sistema tiene dos ecuaciones proporcionales.

En la figura 1 se da el listado del programa y en la 2 un ejemplo de aplicación del mismo. Al final del ejemplo el ordenador debe dar:

¿DIVISION BY ZERO
ERROR IN 540
READY. ■

```
************************
SISTEMAS DE ECUACIONES
************************
ESTE PROGRAMA CALCULA
    LAS SOLUCIONES
   DE UN SISTEMA DE
     N ECUACIONES
         CON
    N INCOGNITAS
======================
NO. DE INCOGNITAS
? 3

COEFICIENTES DE

ECUACION  1
? 2
? 1
? 2

ECUACION  2
? 3
?-1
?-1

ECUACION  3
? 1
? 5
? 6

TERM.INDEPENDIENTES
? 2
? 2
? 5

         SOLUCIONES

A= 1

B= 2

C=-1

DESEA CONTINUAR(S/N)

NO. DE INCOGNITAS
? 6

COEFICIENTES DE

ECUACION  1
? 1
? 1
? 1
? 1
? 1
? 1

ECUACION  2
? 1
? 2
?-1
? 1
? 3
?-1

ECUACION  3
? 2
?-1
? 2
?-1
? 1
? 3

ECUACION  4
? 3
? 2
?-5
? 4
?-2
?-1

ECUACION  5
? 1
? 0
?-1
? 2
? 1
?-1

ECUACION  6
? 2
? 2
? 2
?-3
? 1
?-2

TERM.INDEPENDIENTES
? 2
? 5
? 9
?-22
? 0
? 15

         SOLUCIONES

A= 1.21782178

B=-1.43168317

C= 2.22970297

D=-1.71683168

E= 3.07326733

F=-1.37227723

DESEA CONTINUAR(S/N)

NO. DE INCOGNITAS
? 4

COEFICIENTES DE

ECUACION  1
? 13.6
? 15.42
? 12.04
?-8.89

ECUACION  2
? 7.81
?-5.43
?-12.62
?-1.86

ECUACION  3
? 28.6
?-85.4
? 22.52
? 2.01

ECUACION  4
? 2.25
?-2.12
? 3.75
?-5.55

TERM.INDEPENDIENTES
?-6053.88
?-37796.65
? 426098.65
?-17360.56

         SOLUCIONES

A= 3554.99904

B=-2301.99926

C= 4894.99926

D= 8755.99884

DESEA CONTINUAR(S/N)

NO. DE INCOGNITAS
? 3

COEFICIENTES DE

ECUACION  1
? 1
? 2
?-1

ECUACION  2
? 2
? 4
?-2

ECUACION  3
? 3
?-1
? 1

TERM.INDEPENDIENTES
? 2
? 4
? 4
```

(Fig. 2)

CORREO ABIERTO

Los lectores que se sientan impulsados a plantear una consulta que pueda aclarar sus dudas y —¿por qué no?— la de otros muchos, pueden manifestarla por escrito remitiéndola a: CORREO ABIERTO - «Club Commodore» - Calle Taquigrafo Serra, n.º 7, 5.ª planta - Barcelona-29. Pondremos todas nuestras luces a su disposición para las aclaraciones. ¡Que nadie se quede con sus dudas, cavilando en solitario! — P. MASATS

F. Gutiérrez, de Málaga, nos pregunta: ¿Existe la posibilidad de que un programa «cargue» a otro?

Respuesta: En cuanto a este problema nos remitimos al artículo «GRANDES PROGRAMAS EN POCA MEMORIA» publicado en el número 2 de CLUB COMMODORE donde se da solución a este asunto. En todo caso el ejemplo para el VIC quedaría:

```
 5 REM PROGRAMA 1
10 POKE45, PEEK(174) : POKE46,
   PEEK(175) : CLR
20 FORI=1TO10 : PRINT "PRG 1" :
   NEXT
30 POKE45, PEEK(55) : POKE46,
   PEEK(56) : LOAD " PRG 2 ", 8
```

M. Pérez, de Barcelona, nos pregunta: ¿Se pueden dibujar piezas de ajedrez en el VIC?

Respuesta: De hecho para esto (entre otras cosas) sirve el editor de caracteres publicado en nuestro número 3 (en las ocho páginas que reciben solamente nuestros suscriptores).

J. Ameller, de Calatayud, nos hace dos consultas: ¿Cómo se puede programar un menú, teniendo en cuenta que cada opción cargue y ejecute automáticamente un programa desde el disco?

Respuesta: Esto se puede obtener mediante una ligera modificación del programa que carga a otro desde el disco. Simplemente (ver más arriba) haciendo que la línea 30 ejecute un LOAD de PRG 2, PRG 3 ó PRG 4 dependiendo del valor de una variable.

¿Cómo se pueden editar cifras numéricas en la impresora o en el TV?

Respuesta: Por la manera de plantear la cuestión parece que, en parte, queda contestada en el artículo «AYUDA PARA CONTABLES». Sólo hay que añadir que una vez elaborada la cadena alfanumérica ésta puede ser impresa sin más complicaciones.

L. Casi, de Barcelona, nos formula varias preguntas: ¿Cómo sería el programa que, después de una partida de un juego cualquiera, exhibiese la tabla de los 10 mejores clasificados, sin que en una partida posterior se borrasen?

Respuesta: La manera más fácil consiste en que el programa no tenga que hacer un nuevo RUN cada vez que empieza una nueva partida. Así, en dos matrices unidimensionales de diez elementos, se almacenaría la información; en una numérica, los resultados y en la otra —alfabética— los nombres. Otro procedimiento consiste en grabar en cinta los resultados al final de una partida para su posterior utilización. Véase el manual del usuario del VIC Pág. 109 Apéndice B para más detalles de manejo de información con el cassette.

¿Cuándo aparecerá a la venta la segunda parte del curso de introducción al BASIC?

Respuesta: En estos momentos (Diciembre del 82) se está terminando el proceso de traducción y entregando el primer material a la imprenta. En todo caso, cuando esté disponible se anunciará (de forma silenciosa pero con letras GRANDES) en esta Revista.

¿Dónde puedo encontrar información sobre programas para el VIC?

Respuesta: Obviamente, en esta Revista, que está dedicada a las actividades que se realicen en torno a los ORDENADORES PERSONALES de COMMODORE en general y de forma especial al VIC-20. No resistimos la tentación de insistir, una vez MÁS, en nuestro interés en publicar todas aquellas informaciones sobre estos temas que lleguen a nuestras manos.

* * *

Pregunta: ¿Cuánta memoria queda disponible cuando se trabaja con 32K de RAM?

Respuesta: Se puede disponer de 27,5K.

Pregunta: ¿Qué versión de DOS es la del disco del VIC-20?

Respuesta: El sistema operativo del disco (DOS) del VIC-1540 es la versión standard COMMODORE 2.6.

Pregunta: ¿Cómo pueden pasarse programas de disco a cinta y viceversa?

Respuesta: La única manera de hacer esta transferencia es cargando el programa en memoria con un LOAD y grabándolo con un SAVE.

MAPA DE MEMORIA BASIC 1

000	$4C constant (6502 JMP instruction)
001-002	USR function address lo, hi

Terminal I/O maintenance

003	Active I/O channel #
004	Nulls to print for CRLF (unused).
005	Column Basic is printing next
006	Terminal width (unused).
007	Limit for scanning source colmns (unused)
008	Line number before storage buffer. (integer address from Basic)
009	$2C constant (special comma for INPUT process).
010-089	BASIC INPUT buffer (80 bytes).
090	General counter for BASIC. (search char ':' or endline)
091	$00 used as delimeter (scan between quotes flag).
092	General counter for BASIC. input buffer pointer.
093	Flag to remember dimensioned variables. 1st char of name .
094	Flag for variable type: 0=numeric; 1=string.
095	Flag for integer type: 80=integer; 00=floating point.
096	Flag to crunch reserved words (protects "& remark).
097	Flag which allows subscripts in syntax.
098	Flags INPUT or READ: 0=Input; 64=Get; 152=Read.
099	Flag sign of TAN.
100	Flag to suppress OUTPUT (+normal;-suppressed).
101	Index to next available descriptor.
102-103	Pointer to last string temporary lo; hi.
104-111	Table of double byte descriptors which point to variables.
112-113	Indirect index #1 lo; hi.
114-115	Indirect index #2 lo; hi.
116-121	Pseudo register for function operands.

Data storage maintenance

122-123	Pointer to start of BASIC text area lo; hi type
124-125	Pointer to start of variables lo; hi byte.
126-127	Pointer to array table lo; hi byte.
128-129	Pointer to end of variables lo; hi byte.
130-131	Pointer to start of strings lo; hi byte.
132-133	Pointer to top of string space lo; hi byte.
134-135	Highest RAM adr lo;hi byte.
136-137	Current line being executed. A zero in 136 means statment executed in a direct command.
138-139	Line # for continue command lo; hi.
140-141	Pointer to next STMNT to execute lo; hi.
142-143	Data line # for errors lo; hi.
144-145	Data statment pointer lo; hi.(145-memory address of data line)

Expression evaluation

146-147	Source of INPUT lo; hi.
148-149	Current variable name.
150-151	Pointer to variable in memory lo; hi.
152-153	Pointer to variable referred to in current FOR-NEXT
154-155	Pointer to current operator in table lo; hi.
156	Special mask for current operator.
157-158	Pointer for function definition lo; hi.
159-160	Pointer to a string descriptor lo; hi.
161	Length of a string of above string.
162	Constant used by garbage collect routine.(3or7 for grbg clct)
163	$4C constant (6502 JMP inst).
164-165	Vector for function dispatch lo; hi.
166-171	Floating accumulator # 3
172-173	Block transfer pointer # 1 lo;hi.
174-175	Block transfer pointer # 2 lo; hi.
176-181	Floating accumulator # 1(FAC#1)(USR function evaluated here).
182	Duplicate copy of sign of mantissa of FAC # 1.
183	Counter for # of bits to shaft to normalize FAC # 1.
184-189	Floating accumulator # 2.(FAC#2)
190	Overflow byte for floating argument.
191	Duplicate copy of sign of mantissa.
192-193	Pointer to ASCII rep of FAC in conversion routine lo; hi.

RAM subroutines

194-199	CHARGOT RAM code. Gets next character from BASIC text.
200	CHARGOT RAM code regets current characters.
201-202	Pointer to source text lo; hi.
203-223	Next random number in storage

OS page zero storage

224-225	Pointer to start of line cursor loc lo; hi.
226	Column positionof cursor.(0-79).
227-228	General purpose start address indirect lo; hi.
229-233	General purpose end address direct lo; hi.
234	Flag for quote mode on/off.
235	timer 1 interrupt sttus: 0=disabled
236	EOT character received
237	character error received
238	current file name length.
239	Current logical file number.
240	Current primary address.
241-242	Current secondary address.(241 device no; 242 max line length)
243-244	Pointer to start of current tape buffer lo; hi.
245	Current screen line #.
246	Data temporary for I/O.
247-248	Pointer to start loc for O.S. lo-hi.(tape start adress/pointer)
249-250	Pointer to current file name lo; hi.
251-254	Tape variable storage.
255	Overflow byte BASIC uses when doing FAC to ACIII conversions.

Page 1

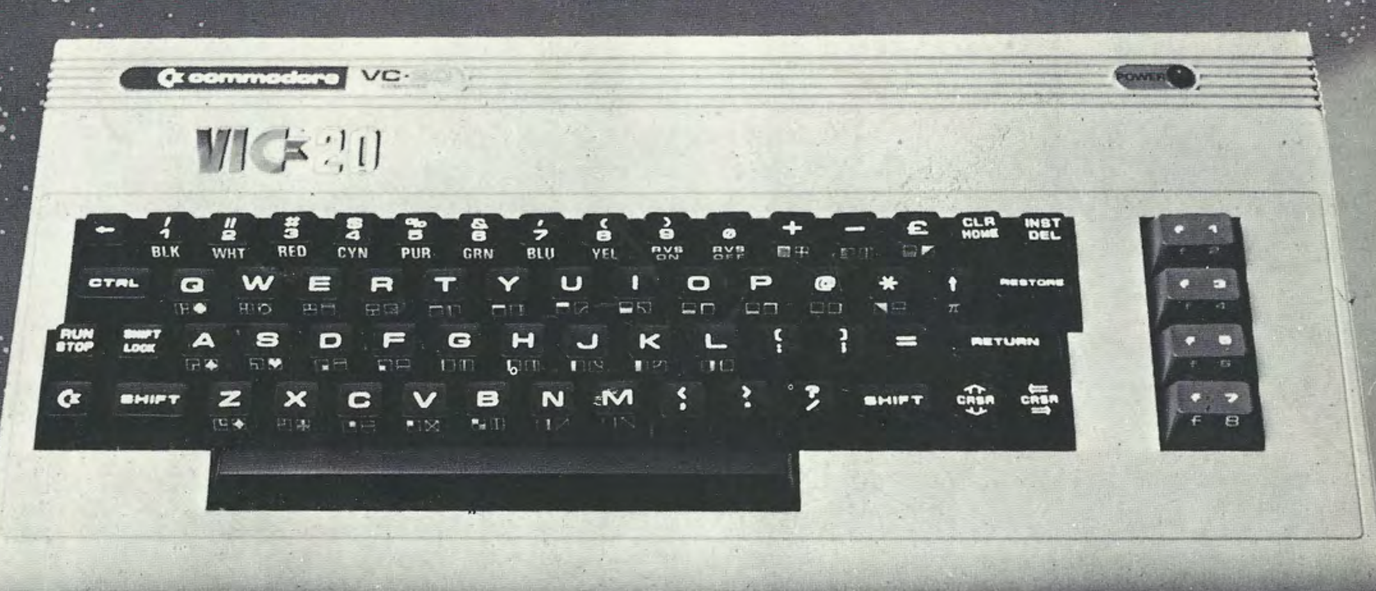
62 bytes on bottom are used for error correction in tape reads. Also, buffer for ASCII when Basic is expanding the FAC into a printable number. The rest of page 1 is used for storage of BASIC GOSUB and FOR NEXT context and hardware stack for the machine.

Page 2

512-514	24-hour clock in 1/60 secs
515	Matrix co-ordinates of last key down (row/col; 255=no key)
516	Shift key status: 0=no shift; 1=shift
517-518	Correction factor for clock, LSB, MSB
519-520	Interrupt driver flag for cassette # 1,switches; # 2 switches (519 for cassette#1 on; 520 for cassette#2 on)
521	Keyswitch PIA duplicate of 59910
522	timing constant buffer
523	Flag # means verify not load into memory.
524	I/O status byte.
525	Index into keystroke buffer.
526	Flag to indicate reverse-field on.
527-536	Interrupt driven key stroke buffer.
537-538	IRQ RAM VECTOR lo; hi.
539-540	BRK instruction RAM VECTOR lo; hi.
541	(IEEE mode)
542	(end of line for input pointer; # characters on screen line)
543	?
544-545	(cursor log row/col, used in input routines)
546	(PBD image for tape I/O)
547	Keyboard input code.
548	Blink cursor flag.
549	Count down to flip cursor. Cursor blink duration.
550	Screen value of input character when cursor moves on.
551	Flag for cursor on/off.
552	(EOT bit received,tape write)
553-577	Table of LSB of start address of video display lines (25).
578-587	Table of logical addresses.
588-597	Table of primary addresses.
598-609	Table of secondary addresses.
608	Input from screen/keyboard flag. 0=keyboard; 1=screen.
610	Index into LA,FA,SA, tables
611	Default input device #.
612	Default output device #.
613	Computation of parity on cassette write.
615-615	?
616	Tape buffer item counter.
617-619	?
620	Serial bit count.
621	Count of redundant tape blocks.
622	?
623	(cycle counter,flip for every bit coming from tape)
624	Count down synchronization or cassette write.
625-626	Index next character in/out tape buffer # 1; # 2.
627	Countdown synchronization on cassette header.
628	Flag to indicate bit/byte error.
629	Flag to indicate tape routine reading shorts.
630-631	Index to addresses to correct on tape read pass 1; pass.
632	Flag for cassette read-tells current function-countdown,read,etc
633	Count of seconds of shorts to write before data.
634-825	Buffer for cassette # 1 (192 bytes)
836-1017	Buffer for cassette # 2 (192 bytes)
1018-1023	Unused.

VIC-20

EL ORDENADOR PERSONAL AMPLIABLE CON COLOR Y SONIDO.



49.500 Ptas.
COLOR-SONIDO

Así es el VIC-20

- Lenguaje BASIC extendido.
- Sistema operativo COMMODORE.
- 5 K RAM ampliable a 32 K.
- 16 colores, 4 generadores de sonido.
- 66 caracteres gráficos.
- Periféricos disponibles:
  - Cassette.
  - Impresora de agujas.
  - Unidad de disco de 170 K.

Así hace las cosas el VIC-20

- Enseña informática.
- Efectúa todo tipo de cálculos matemáticos.
- Realiza funciones docentes.
- Se encarga de múltiples tareas profesionales.
- Proporciona divertidos momentos de ocio.
- Ayuda a planificar labores domésticas.
- Hace todas las aplicaciones que Vd. imagine.



GRATIS

Con la adquisición de su VIC-20 recibirá además:
- MANUAL DEL USUARIO.
- INTRODUCCION AL LENGUAJE DE PROGRAMACION BASIC.
- Y 17 PROGRAMAS DE PRACTICAS (en dos cassettes).



commodore COMPUTER

Distribuidor exclusivo para España:

Microelectrónica y Control, S.A.
Taquígrafo Serra, 7 5.°. Barcelona-29
Princesa, 47 3.° G. Madrid-8

De venta en tiendas especializadas.

SLOGAN